

3. COLA DE PRIORIDAD

DEFINICION (I)

✱ **Conjunto de elementos ordenados con las operaciones:**

Crear () \rightarrow ColaPrioridad

EsVacio () \rightarrow Boolean

Insertar (ColaPrioridad, Item) \rightarrow ColaPrioridad

BorrarMínimo (ColaPrioridad) \rightarrow ColaPrioridad

BorrarMáximo (ColaPrioridad) \rightarrow ColaPrioridad

Búsqueda (ColaPrioridad, Item) \rightarrow Boolean

Cardinalidad (ColaPrioridad) \rightarrow Natural

Copiar (ColaPrioridad) \rightarrow ColaPrioridad

Mínimo (ColaPrioridad) \rightarrow Item

Máximo (ColaPrioridad) \rightarrow Item

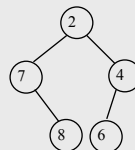
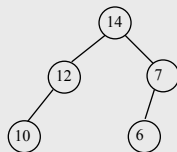
1

3. COLA DE PRIORIDAD

DEFINICION (II)

✱ **Árbol Mínimo (Máximo):**

Árbol en el que la etiqueta de cada nodo es menor (mayor) que la de los hijos.



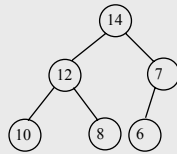
2

3. COLA DE PRIORIDAD

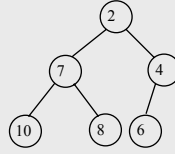
DEFINICION (III)

Heap Mínimo (Máximo):

Árbol binario completo en que además es ARBOL MINIMO o MAXIMO.



HEAP MAXIMO



HEAP MINIMO

3. COLA DE PRIORIDAD

DEFINICION (IV)

Implementación Cola Prioridad:

■ LISTA DESORDENADA:

INSERCIÓN: $O(1)$

BORRADO: $O(n)$

■ LISTA ORDENADA: ascendente o descendente.

INSERCIÓN: $O(n)$

BORRADO: $O(1)$

■ ÁRBOL BINARIO DE BUSQUEDA:

INSERCIÓN: $O(n)$

BORRADO: $O(n)$

3. COLA DE PRIORIDAD

DEFINICION (V)

Implementacion Cola Prioridad:

HEAP o MONTICULO:


INSERCIÓN:	$O(\log n)$
BORRADO:	$O(\log n)$

5

3.1. HEAP MAXIMO (MINIMO)

INSERCIÓN

METODO:

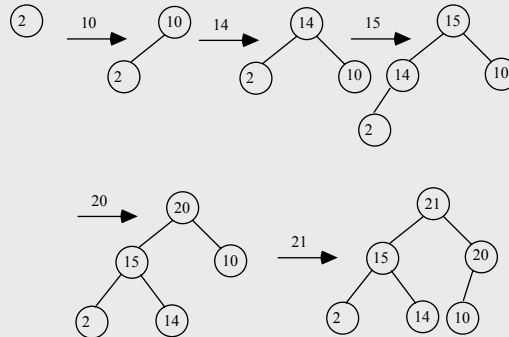
- 1.- Insertar en la posición correspondiente para que siga siendo un árbol completo.
- 2.- Reorganizar para que cumpla las condiciones del HEAP:
 -  Comparar con el nodo padre: si no cumple las condiciones del árbol mínimo/máximo, entonces intercambiar ambos.

6

3.1. HEAP MAXIMO (MINIMO)

INSERCIÓN. EJEMPLO

✱ Insertar: 2, 10, 14, 15, 20 y 21 en un heap máximo inicialmente vacío



7

3.1. HEAP MAXIMO (MINIMO)

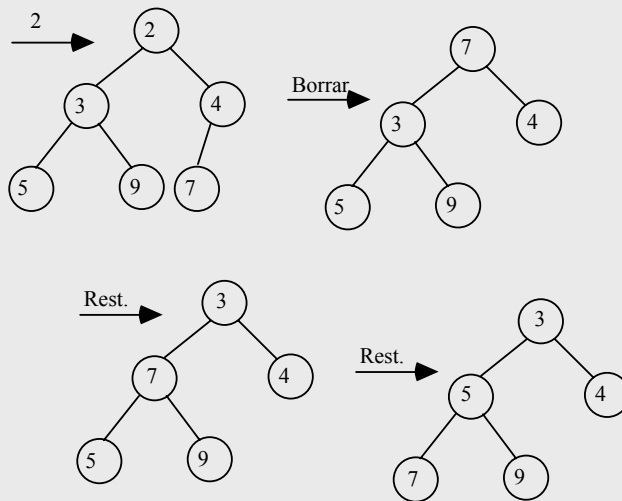
BORRADO.

✱ **METODO:**

- Se sustituye la raíz con el elemento más a la derecha en el nivel de las hojas
- Mientras no sea un HEAP se hunde ese elemento sustituyéndolo con el más pequeño (montículo mínimo) o el mayor (montículo máximo) de sus hijos

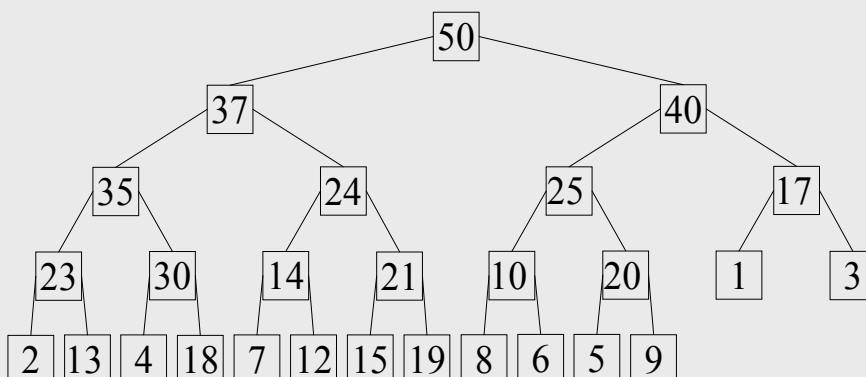
8

3.1. HEAP MAXIMO (MINIMO)



3.1. HEAP MAXIMO (MINIMO)

 Realiza dos borrados sobre el siguiente montículo máximo.



3.1. HEAP MAXIMO (MINIMO)

INSERCIÓN. EJEMPLO II

🚧 Sobre el resultado anterior, realiza las inserciones: 60, 36

11

3.1. HEAP MAXIMO (MINIMO)

REPRESENTACIÓN

ENLAZADA: problema en inserción al necesitar realizar recorridos ascendentes.

SECUENCIAL (en un vector):

Hijos de $p[i]$ son $p[2 \cdot i]$ y $p[2 \cdot i + 1]$. Padre de $p[i]$ es $p[i \text{ DIV } 2]$ con DIV la división entera

12

3.1. HEAP MAXIMO (MINIMO)

APLICACIÓN HEAPSORT

- # Algoritmo de ordenación de un vector de elementos
- # **METODO:**
 - 1) Insertar los elementos en un HEAP
 - 2) Realizar borrados de la raíz del HEAP
- # **IMPLEMENTACIÓN (UN SÓLO VECTOR):**
 - 1) Dejar parte izquierda del vector para el HEAP, y parte derecha para los elementos todavía no insertados.
 - 2) Borrar la raíz del HEAP llevándola a la parte derecha del vector.
- # **COMPLEJIDAD:**

$O(n \log n)$

13

3.1. HEAP MAXIMO (MINIMO)

HEAPSORT. EJERCICIO

- # Ordenar el vector 5 2 7 3 1 usando un heap máximo
- # Ordenar el vector 9 5 7 4 8 6 2 1 usando un heap mínimo

14

3.2. COLA DE PRIORIDAD DOBLE (DEAP)

DEFINICIÓN (I)

- ✦ Cola de Prioridad doble: cola de prioridad en la que se soporta la operación de borrado de la clave máxima y mínima.
- ✦ DEAP: Es un Heap que soporta las operaciones de cola de prioridad doble.

DEFINICION: es un árbol binario completo el cual o es vacío o satisface las siguientes propiedades:

- 1) La raíz no contiene elementos
- 2) El subárbol izquierdo es un HEAP mínimo
- 3) El subárbol derecho es un HEAP máximo
- 4) Si el subárbol derecho no es vacío:

- Sea “i” cualquier nodo del subárbol izquierdo.

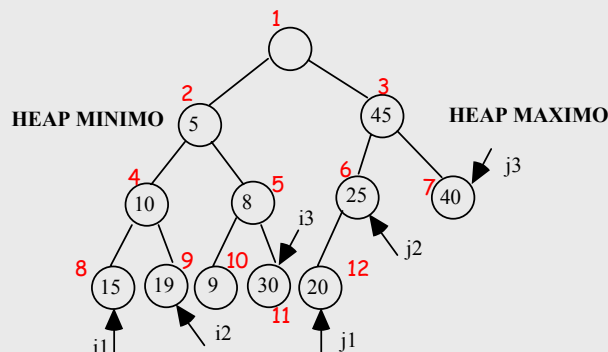
- Sea “j” el nodo correspondiente en el subárbol derecho. Si “j” no existe, entonces sea el nodo del subárbol derecho que corresponde al padre de “i”.

- Entonces: clave (i) < clave (j)

15

3.2. COLA DE PRIORIDAD DOBLE (DEAP)

DEFINICIÓN (II)



16

3.2. COLA DE PRIORIDAD DOBLE (DEAP)

IMPLEMENTACIÓN

Igual que en un HEAP, sólo que la primera posición no se utilizará.

$$j = i + 2^{(\log_2 i) - 1} \quad () \text{ parte entera}$$

Si $j > n$ Entonces $j = j \text{ DIV } 2$

Ejemplo:

simétrico de i1 (i=8). $j = 8 + 2^{(\log_2 8) - 1} = 8 + 2^2 = \underline{12}$

simétrico de i2 (i=9). $j = 9 + 2^{(\log_2 9) - 1} = 9 + 2^2 = 13$. como $j > n \rightarrow j = 13 \text{ DIV } 2 = \underline{6}$

17

3.2. COLA DE PRIORIDAD DOBLE (DEAP)

INSERCIÓN

- 1) Se inserta el elemento en el siguiente índice del árbol completo
- 2) Se compara el nodo insertado con el nodo simétrico correspondiente, realizando el intercambio en caso que no se cumpla la condición 4 de la definición del DEAP:

$$i = n - 2^{(\log_2 n) - 1} \quad () \text{ parte entera}$$

- 3) Actualizar el HEAP mediante el proceso de “ascensión” del elemento insertado.

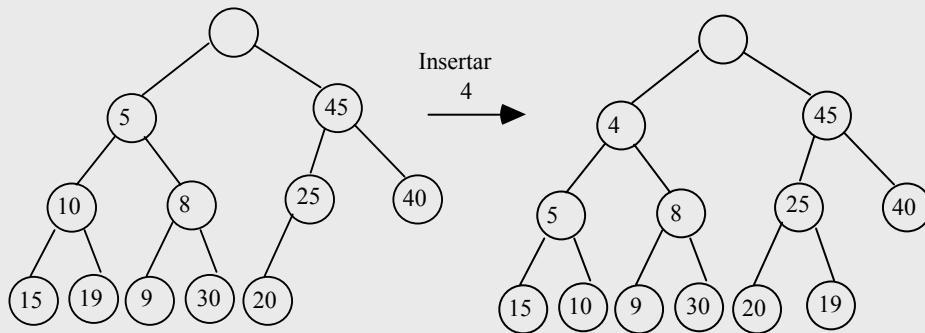
Ejemplo:

simétrico de j1 (j=12). $i = 12 - 2^{(\log_2 12) - 1} = 12 - 2^2 = \underline{8}$

18

3.2. COLA DE PRIORIDAD DOBLE (DEAP)

INSERCIÓN



19

3.2. COLA DE PRIORIDAD DOBLE (DEAP)

INSERCIÓN

✦ Sobre el DEAP anterior insertar: 35, 7, 50, 17, 12, 27, 55

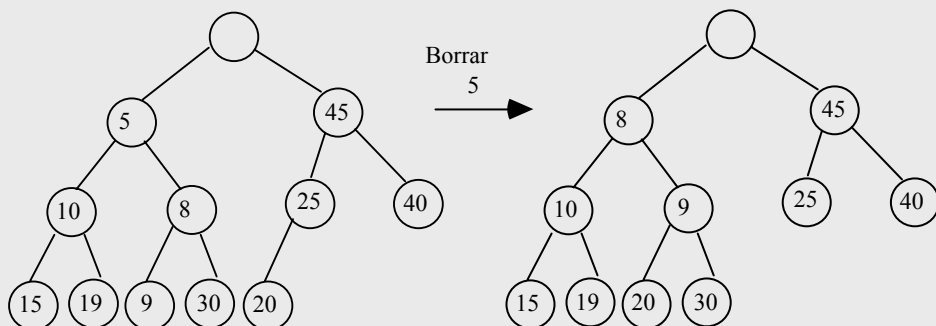
20

3.2. COLA DE PRIORIDAD DOBLE (DEAP) BORRADO

- 1) Intercambiar la raíz a borrar del HEAP con el elemento más a la derecha del último nivel del árbol, y borrar éste.
- 2) Actualizar el HEAP, “hundiendo” la clave intercambiada.
- 3) Comprobar que la clave intercambiada no incumpla la condición del DEAP con su correspondiente nodo simétrico
- 4) Actualizar el montículo en el que quede la clave intercambiada

21

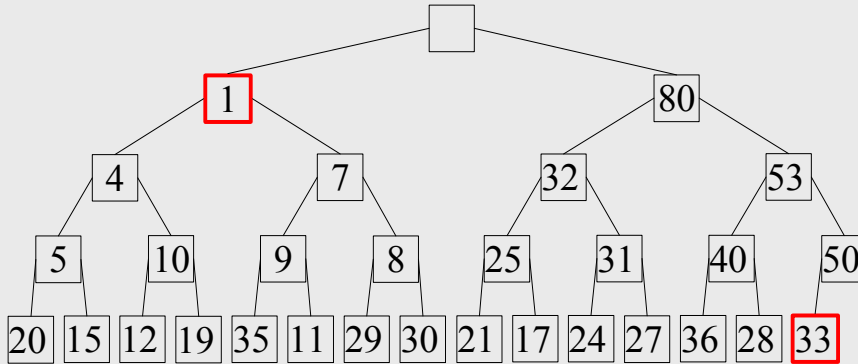
3.2. COLA DE PRIORIDAD DOBLE (DEAP) BORRADO



22

3.2. COLA DE PRIORIDAD DOBLE (DEAP) BORRADO

MONTÍCULO DOBLE: Borrar los elementos mínimo y máximo de forma sucesiva.



23

3.3. ARBOLES LEFTIST DEFINICIÓN (I)

- ✦ Se utilizan en las colas de prioridad que implementen la operación: **Combinar** (ColaPrioridad, ColaPrioridad) → ColaPrioridad
- ✦ **CAMINO MINIMO (X):** longitud del camino más corto desde X hasta un árbol vacío.

$$\text{CMINIMO}(X) = \begin{cases} 0 & \text{si } X \text{ es un árbol vacío} \\ 1 + \min(\text{CMIN}(\text{HijoIzq}(X)), \text{CMIN}(\text{HijoDer}(X))) & \text{en otro caso} \end{cases}$$
- ✦ **ARBOL LEFTIST:** árbol binario tal que si no es vacío:

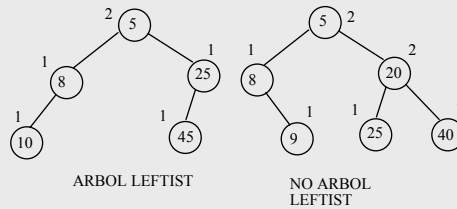
$$\text{CMIN}(\text{HijoIzq}(x)) \geq \text{CMIN}(\text{HijoDer}(x)) \text{ Para todo } x \text{ no vacío}$$

24

3.3. ARBOLES LEFTIST

DEFINICIÓN (II)

ARBOL LEFTIST MINIMO (MAXIMO): árbol leftist tal que la clave de cada nodo es menor (mayor) que la de sus hijos.



25

3.3. ARBOLES LEFTIST

OPERACIÓN (I)

COMBINAR (A, B):

1) Se toma como árbol resultado de la combinación aquel con etiqueta mínima. El subárbol izquierdo no se toca.

2) Sobre el subárbol derecho se pone el resultado de combinar el subárbol derecho de éste y el árbol que tenía una etiqueta mayor, controlando que:

- Una vez enraizado el padre, la propiedad de leftist: intercambiar los dos hijos.

- En cada nuevo subárbol creado actualizar el camino mínimo para las raíces.

26

3.3. ARBOLES LEFTIST

OPERACIÓN (II)

ALGORITMO CombinarMinimo

ENTRADA: A, B : LEFTIST;

SALIDA: A, B : LEFTIST;

METODO

Si ESVACIO (A)

Mover (A, B)

Sino

UnionMin (A, B)

27

3.3. ARBOLES LEFTIST

OPERACIÓN (III)

ALGORITMO UnionMin

ENTRADA/SALIDA : i, j; Iterador;

VAR : k: Iterador; C: Leftist;

METODO

Si dato (i) > dato (j)

k = C;

Mover (k, i); Mover (i, j); Mover (j, k)

fsi

Si EsVacio (HijoDer (i))

Mover (HijoDer (i), j);

Sino

UnionMin (HijoDer (i), j)

fsi

Si EsVacio (HijoIzq (i))

Mover (HijoIzq (i), HijoDer (i));

Sino

Si CMIN (HijoIzq(i)) < CMIN (HijoDer (i))

Mover (k, HijoIzq (i));

Mover (HijoIzq (i), HijoDer (i));

Mover (HijoDer (i), k)

fsi

fsi

Si EsVacio (HijoDer (i))

CMIN (i) = 1;

Sino

CMIN (i) = CMIN (HijoDer (i)) + 1;

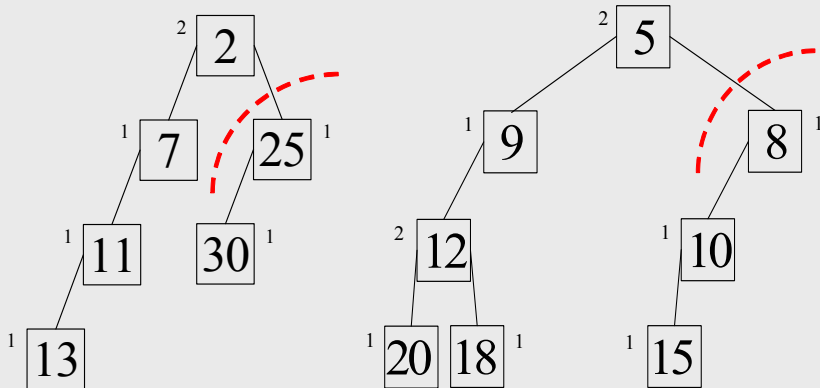
fsi

finMETODO

28

3.3. ARBOLES LEFTIST

COMBINAR



29

3.3. ARBOLES LEFTIST

INSERTAR

✳ **Insertar:** 4, 8, 6, 3, 1, 12 en un leftist mínimo inicialmente vacío.

30

3.3. ARBOLES LEFTIST

BORRAR

Realiza tres borrados sobre el siguiente árbol leftist mínimo:

